全国计算机自学考试全程过关必备丛书

数据结构导论习题与真题解析

邹华跃　张　华　等编著

中国水利水电出版社

前　言

《数据结构导论》是计算机及应用专业中的一门专业基础课程，在计算机学科中起着承前启后的作用，在计算机技术的各个领域中也有着广泛的应用。因此，它是一门重要的专业基础课程。

数据结构涉及数据的组织、存储以及运算的一般方法，其原理及算法较为抽象，对刚刚接触计算机学科的考生来讲，了解与掌握其中的原理显得尤为困难。有些章节的内容自学起来难度特别大，犹如读天书一样。在解答习题时，往往也感到无从下手。作者借编写本书的机会，对多年来的教学实践进行了系统的总结，将自己对课程的认识、体会、感想融入到本书中，并尽可能从考生的角度出发，对难以理解的原理及算法进行了通俗化处理，以求高度概括，易学易懂。作者的愿望是，通过对本书的阅读，能让考生把握本课程的主线，加深对基本概念的理解，掌握求解数据结构问题的思路与方法，提高分析与解决问题的能力，直至最终顺利通过自学考试。

本书按照全国高等教育自学考试委员会颁布的《数据结构导论自学考试大纲》的要求，以考核知识点与考核要求为主线，再结合作者多年来的教学经验，从整体上对本书的结构及内容进行规划、编排，全书分为三篇，它们在内容上各有特色，相互配合，彼此补充。其中：

本章知识点讲解部分：最能体现作者对本课程的认识与理解，是作者多年来教学工作的结晶，其特色是通过简洁、通俗的语言对难以理解的原理、算法进行了归纳与总结。

配套教材习题分析与解答部分：对指定教材《数据结构导论》（经济科学出版社，陈小平主编）中的所有习题进行了详细的分析与解答，使考生能够知其然也知其所以然。

典型题分析与解答部分：反映了作者对自考大纲的准确把握。这里不注重题量的多少，但是在题目的典型性、题型、难易程度等方面，力求与自考大纲贴近，以增强考生的应试能力。

往年自考真题分析与解答部分：由于往年自考真题对考生来讲有着重要的参考价值，所以作者尽可能收集了自 2000 年以来的自考试卷真题，并分章节作了详细的分析与解答，以便考生有针对性地复习与自我测试。

最新自考真卷分析与解答部分：这部分提供的最新自考真卷用于考生临考前的自查自检，以便大家能从中体会到自考试卷的题型、题量、考核点及试题的难易程度。

由此，我们有理由相信，本书能够成为考生的良师益友，是考前辅导的好帮手。

本书由邹华跃、张华编写，参加本书资料收集、整理、编写的还有汪曙华、邹长青、高洪斌、俞冬梅、郑德忠、胡芳等同志。中国水利水电出版社的杨庆川、张玉玲等同志为本书的出版花费了大量的精力，在此表示衷心的感谢。

本书涉及的题量较大，在分析、解答上难免会存在不够完整或疏漏之处，恳请广大读者批评指正。

作者

2004 年 5 月

目　　录

第三部分 最新自考真题试卷解析

第一部分　知识点与配套教材习题解析

本篇按自考大纲和指定教材的顺序划分章节。每章都围绕相关内容提炼出考核知识点，对每个知识点不像教材那样详细讲解，而是给出结论性的提示，这是本书最具特色的地方，也最能体现作者对本课程的理解和体会。考生只需掌握这些知识点，再结合第二篇中的典型题解，便可迅速了解相关内容的考题形式、深度、广度和难度。尽管同一问题可能会以不同形式的题型出现，但这不过是一种命题技巧而已。

本篇的第二个内容是配套教材中的习题解析。教材中的习题都紧扣自考大纲和各章节的内容，因此对考生来讲，钻研这些习题的解题方法就成了学习中的重中之重。作者结合十多年的教学经验和考生经常遇到的问题，对配套教材中每章后的习题进行了详细的分析与解答，让考生真正做到知其然也知其所以然。

第 1 章 概论

计算机是一种对信息进行加工处理的机器。研究计算机，主要是研究计算机信息的组织、加工和处理方法。因此，对计算机应用专业的学生来说，理解信息的组织、加工、处理等方面的基本概念显得尤为重要。本章讲述数据结构的概念、实现方法及算法的时间复杂性分析。

1.1 知识点

1. 数据、数据元素和数据项的概念

数据是指所有能输入计算机中并能被计算机加工、处理的符号的集合。它是信息的载体，其含义极其广范，诸如数、符号、字体、图形、声音等都可以看作是数据。因此，在概念上不同于大家平常理解的“数”的概念。

数据元素是数据的基本单位，通常具有完整、确定的实际意义，并被当作运算的基本单位。比如数据库中的库文件可以被看作数据，而其中的一条条记录就对应于一个个数据元素，它们有实际意义，也是数据库操作的基本单位。在后续章节中，数据元素又被称为元素、记录、结点、顶点，以便形象化地描述。

数据项是数据不可再分的最小标识单位，它不具有完整的实际意义，通常仅反映数据元素某一方面的属性。在很多场合下，数据项又被称为字段、数据域。

综合起来看，数据、数据元素和数据项反映出了数据组织的 3 个层次，它们之间的关系是数据可由若干个数据元素构成，而数据元素又可由若干个数据项构成。

2. 数据的逻辑结构

通常数据非常复杂，其中的数据元素之间可能存在着各种各样的关系。比如工厂里的工人之间，可能存在着上下级关系、血源关系、同乡关系，同学关系等，用数据结构的眼光来看，所有这些关系都可以抽象为数据元素之间的逻辑关系，而数据元素之间的逻辑关系的整体就构成了数据的逻辑结构。由此可知，数据的逻辑结构实际上就是数据的组织形式，它反映的是数据元素之间的一种关联方式，或称“邻接关系”。

数据的逻辑结构分为 4 种基本类型：集合、线性结构、树形结构和图状结构。

（1）集合。如果数据中的数据元素之间不存在任何逻辑关系，则称其逻辑结构为集合。也就是说，集合是指数据元素之间“没有关系的关系”的整体，因此集合的组织形式是最松散的，不受任何制约。

（2）线性结构。在线性结构中，数据元素之间是呈线性排列的。比如，张三与李四是小学同学，李四与王五是中学同学，王五与孙六是大学同学，这样张三、李四、王五、孙六之间的“同学”关系就是一种线性关系。线性结构的特点是有头有尾，相互之间次序

分明。

（3）树形结构。有些数据元素之间的关系呈现出一种层次特性，比如同事之间的上下级关系，家族中的血源关系等。从形态上看，这种层次关系像是一棵倒置的树，所以把这种逻辑结构形象地称为树形结构。树形结构的特点就是它的层次性。

（4）图状结构。图状结构是 4 种基本逻辑结构中最复杂的一种。因为在这种逻辑结构中，任意两个数据元素之间都有可能发生关联，而且这种关联没有任何规律可寻。比如城市之间的公路网或通讯网就是一种图状结构。

关于逻辑结构，教材上特别指出以下几点，需要引起注意：

- 逻辑结构与数据元素本身的形式、内容无关。
- 逻辑结构与数据元素的相对位置无关。
- 逻辑结构与所含数据元素的个数无关。
- 逻辑结构与数据的存储无关，它是独立于计算机的。

3. 运算的概念

数据的运算是指对数据施加的操作。最常用的运算有检索、插入、删除、更新、排序等。这些运算实际上是在抽象的数据上所施加的抽象操作，因为在这里我们只关心这些操作是“做什么”的，而无须考虑“如何做”。

有一点需要明确，数据的运算是定义在数据的逻辑结构上的，因此每一种逻辑结构对应着一个运算的集合。

4. 数据结构的概念

通俗地讲，数据结构是带有关系的数据元素的集合。研究数据结构的目的是为了在计算机中实现对它们的操作，这样就引出两个问题：

（1）数据结构如何在计算机中表示？

（2）在数据结构上定义怎样的操作（运算）？

由此，我们可以知道数据结构包含了 3 个方面的内容：

（1）数据的逻辑结构：反映的是数据元素之间的逻辑关系。

（2）数据的存储结构：即数据结构在计算机内是如何表示的。

（3）数据的运算：定义在逻辑结构上的、对数据元素所施加的操作。

综上所述，数据结构的定义可以描述为：由某种逻辑关系组织起来的一批数据按一定的存储表示方式存储在计算机中，并在这些数据上定义一组运算。

5. 存储结构和运算实现

数据的存储结构是指数据在计算机内的表示，它涉及数据元素的表示及元素之间关系的表示两个方面。数据的存储方式有 4 种，它们的基本思想及特征如下：

（1）顺序存储结构。这种存储方式是把逻辑上相邻的数据元素存储到物理位置上相邻的存储单元里，因此，数据元素之间的逻辑关系就由存储单元的相邻关系来体现。

有两点需要注意：

- 顺序存储结构使用的是一段连续空间，中间不允许有间隙。
- 要“按一定的次序”将元素装入存储器里，这种次序同时也决定了对数据元素的存取方式。在非线性结构的顺序存储结构中，这一点表现得尤为突出。

（2）链式存储结构。在这种存储方式中，数据元素之间的逻辑关系由附加的指针来表示。因此，对数据元素的具体存放位置没有限制，只要在存放一个数据元素的同时存储一个或多个指针，让这些指针指向与本元素有关联的数据元素即可。

（3）索引存储结构。该方法是在存储数据元素的同时建立一张附加的索引表，用索引表中的索引项来指示各数据元素的存储位置或存储位置的区间起点。

（4）散列存储方式。这种存储方式的本质是，将数据元素的关键字带到散列函数中，通过计算得到其存储地址。也就是说，它是通过散列函数在数据元素本身与其存储地址之间建立起直接的联系。

下面简单谈谈运算的实现。前面讲到运算是定义在数据的逻辑结构之上的，这时我们只关心它的功能，即能“做什么”，而运算的实现却是针对数据的存储结构而言的，它的核心问题应该是“怎样做”，即平常我们所说的算法设计。

6. 算法及其描述

通俗地讲，算法就是解决问题的方法和步骤，它是用语言来描述的。一个算法可以用自然语言、标准程序设计语言或伪语言来描述。用自然语言来描述算法非常随意、方便，但是它有一个致命的弱点，表示不严格，会出现二义性。标准程序设计语言恰恰与它相反，表示很严格，不会出现二义性，但是语言规范太死，必须经过培训才会运用。综合上述两类的优缺点，人们想到用一种类语言（又称为伪语言、假语言）来描述算法，使它既能自然、方便地被使用，又具有一定的严格性，不会出现二义性。现在常用的类语言有类 C 语言、类 Pascal 语言。

7. 算法分析

求解同一问题可以有许多不同的算法，那么怎样来衡量算法的好坏呢？这就涉及到算法分析。衡量一个算法是否优异，主要取决于以下 4 点：

（1）能否正确地实现预定的功能。

（2）执行算法所消耗的时间。

（3）执行算法所消耗的存储空间。

（4）算法是否易于理解、编码、调试等。

在本教材中，主要讨论的是算法的时间性能，偶尔也讨论空间性能。一个算法所消耗的时间与算法中每条语句执行的次数（也称为频度）有直接关系。在很多情况下，语句的执行次数又取决于初始数据量 n（称其为问题规模）的多少，而一个算法的时间复杂性就是该算法所求解问题规模 n 的函数。当问题规模 n 趋向于无穷大时，则把时间复杂性的数

量级称为算法的渐近时间复杂性。在算法分析时，经常将渐近时间复杂性简称为时间复杂性，用 T(n)=O(f(n))表示，其中 f(n)是算法中频度最大的那条语句频度的数量级。

在很多算法中，其时间复杂性还与所处理数据的分布状态有关。有时会根据各种可能出现的数据分布状态中最坏的情况来估计算法的最坏时间复杂性；有时也会对数据分布作出某种假设（如等概率），然后估计算法的平均时间复杂性。

上述关于数据分析涉及到 5 个方面，而后续章节中对每一种数据结构的讨论，也是按这 5 个方面（步骤）来展开的：

- 逻辑结构上的特点。
- 定义在逻辑结构上的基本运算。
- 数据在计算机内的表示（即存储结构）。
- 运算在具体存储结构上的实现。
- 对算法的时间性能、空间性能进行评价。

每种数据结构的上述 5 个方面都是密切相关的，而不同数据结构的对应方面也有着相互联系。所以在自学过程中，大家要善于进行比较，找出彼此间的相同点和不同点，这样有助于加深理解并逐渐在头脑中形成一个完整的体系。

1.2　配套教材习题分析与解答

1．简要回答下列问题：

（1）数据与数据元素有什么区别？

（2）为什么说数据元素之间的逻辑关系是数据内部组织的主要方面？

（3）逻辑结构与存储结构是什么关系？

（4）运算与运算的实现是什么关系？有哪些相同点和不同点？

（1）**【解答】**

数据：凡是能被计算机存储、加工的对象通称为数据。

数据元素：它是数据的基本单位，在程序中通常作为一个整体而加以考虑和处理。

两者的区别：一个数据可以由一个或多个数据元素构成。

（2）**【解答】**

所谓逻辑关系是指数据元素之间的关联方式或称“邻接关系”，数据元素不可能是孤立存在的，它们之间总是存在着某种关系，这种数据元素之间逻辑关系的整体称为逻辑结构。一些表面上很不相同的数据可以有相同的逻辑结构，逻辑结构是数据组织的某种“本质性”的东西，所以逻辑关系是数据内部组织的主要方面。

数据的逻辑结构还有以下几点需要特别注意：

1）逻辑结构与数据元素本身的形式、内容无关。

2）逻辑结构与数据元素的相对位置无关。

3）逻辑结构与所含结点的个数无关。

（3）【解答】

逻辑结构反映的是数据元素之间的逻辑关系，而存储结构是数据结构在计算机中的表示，它包括数据元素的表示及其关系的表示。

（4）【解答】

运算是指在逻辑结构上施加的操作，而运算的实现是指一个完成该运算功能的程序。

相同点：运算和运算的实现都能完成对数据的“处理”或某种特定的操作。

不同点：运算只是描述处理功能，不包括处理步骤和方法，而运算实现的核心则是处理步骤。

2．设计求解下列问题的类 C 语言算法，并分析其最坏情况时间复杂性及其量级。

（1）在数组 A[1..n]中查找值为 K 的元素，若找到则输出其位置 i（1≤i≤n），否则输出 0 作为标志。

（2）找出数组 A[1..n]中元素的最大值和次最大值（本小题以数组元素的比较为标准操作）。

（1）【解答】

```
int locate(datatype a[1..n];datatype k)
{  i=n;                                    /*搜索的起点*/
   while (i≥1 && a[i]!=k)
      i--;                                 /*往前搜索*/
   return(i);
}
```

当查找不成功时，总是比较 n+1 次，所以最坏时间复杂性为 n+1，其量级为 T(n)=O(n)。

（2）【解答】

```
void max_smax(datatype a[1..n],max,smax)
{ max=a[1];smax=a[1];                  /*max，smax 分别表示当前最大值和次最大值*/
  for(i=2;i≤n;i++)
   if(a[i]>max)
     { smax=max;
       max=a[i];                       /* 替换最大值*/
     }
   else if (a[i]> smax) smax=a[i];     /* 替换次最大值*/
}
```

为了得到最大值和最小值，必须经过 n-1 次循环比较，所以最坏时间复杂性为 n-1，其量级为 T(n)=O(n)。

3．略。

第 2 章　线性表

线性表是最简单、最常用的一种数据结构。本章的基本内容是：线性表的逻辑结构定义、各种存储结构的描述方法以及在线性表的顺序存储结构和链式存储结构上是如何实现基本运算的。

2.1　知识点

1. 线性结构的概念

线性结构是由 n（n≥0）个结点（数据元素）$a_1,a_2,\cdots,a_n$ 组成的有限序列。其中 a_1 是有限序列的第一个元素，称其为起始结点，a_n 是有限序列的最后一个元素，称其为终端结点。对任意两个相邻的结点 a_i 和 a_{i+1}（$1\leqslant i<n$），由于 a_i 位于 a_{i+1} 前面，并且两者相邻，故称 a_i 是 a_{i+1} 的直接前趋，称 a_{i+1} 是 a_i 的直接后继。

由上述定义可以看出线性结构（非空的线性结构）的基本特征是：

（1）有且仅有一个起始结点 a_1，它没有直接前趋，只有一个直接后继 a_2。

（2）有且仅有一个终端结点 a_n，它没有直接后继，只有一个直接前趋 a_{n-1}。

（3）其余结点 a_i（$2\leqslant i\leqslant n-1$）均有且仅有一个直接前趋 a_{i-1} 和一个直接后继 a_{i+1}。

由此得到，线性结构中结点间的邻接关系实际上就反映出了结点之间的逻辑关系。

2. 线性表的概念

从逻辑上讲，线性表是一种线性结构。其中所含结点的个数 n 称为线性表的长度。

当 n=0 时，线性表为空表，记为（）或Φ。

当 n≠0 时，线性表为非空表，记为（$a_1,a_2,\cdots,a_n$）。

对于线性表，常见的基本算法有以下几种：

（1）INITATE(L)：对线性表 L 进行初始化，使其成为一个空表，即 L=Φ。

（2）LENGTH(L)：求线性表 L 的长度（即所含结点的个数）。

（3）GET(L,i)：当 $1\leqslant i\leqslant n$ 时，读取线性表 L 的第 i 个元素 a_i。

（4）LOCATE(L,X)：在线性表 L 中查找值为 X 的结点。如果查找成功，运算结果为首次找到值为 X 的结点的序号，如果查找失败，运算结果为 0。

（5）INSERT(L,X,i)：在线性表 L 的第 i（$1\leqslant i\leqslant n+1$）个位置上插入值为 X 的新结点，使编号为 i,i+1,i+2,…,n 的结点分别变为编号为 i+1,i+2,…,n+1 的结点。

（6）DELETE(L,i)：删除线性表 L 的第 i（$1\leqslant i\leqslant n$）个结点，使编号为 i+1,i+2,…,n 的结点分别变成编号为 i,i+1,i+2,…,n-1 的结点。

3. 线性表的顺序存储结构——顺序表

线性表最简单、最自然的存储方式是顺序存储方式，它是把线性表中的结点按逻辑次

序依次存放到一组地址连续的存储单元里。它的特点是逻辑相邻的结点其物理位置一定相邻。用这种方法存储的线性表简称为顺序表。

由于高级语言中的向量（一维数组）也是采用顺序存储方式来表示的，所以可以用向量这种数据类型来描述顺序表。教材中给出了顺序表的类型定义描述，从中我们知道顺序表由两部分组成，一部分是用于存放结点的一个一维数组 data，另一部分是用于指示顺序表长度的变量 last。

假设顺序表 L 是一个 splist 类型的变量（见教材），那么可以通过下列方式对两个域进行访问：

L.data[i]里存放的是顺序表中第 i+1 个结点 a_{i+1}，其中 $0 \leqslant i \leqslant n-1$。

L.last 既可以用于表示顺序表的长度，又可以用于指示终端结点 a_n 的存放位置（a_n 的存放单元序号为 L.last-1）。

当顺序表中每个结点占用 l（$l \geqslant 1$）个存储单元，而且已知起始结点 a_1 的第一个单元的地址是 b，则可以通过下列公式求得任一结点 a_i（$1 \leqslant i \leqslant n$）的第一个存储单元的地址 $Loc(a_i)$：

$$Loc(a_i)=b+(i-1)*l$$

也就是说，在顺序表中，每个结点 a_i 的存储地址是该结点在线性表中的位置 i 的线性函数，只要知道 a_1 的起始地址和每个结点的大小，即可在相同时间内求出任一结点的存储地址，故顺序表是一种随机存取结构。

4. 插入、删除和定位运算在顺序表上的实现

（1）插入运算。在顺序表中实现插入运算，其核心步骤是从 $a_n,a_{n-1},\cdots,a_i$ 依次向后移一个位置，腾出第 i 个位置以使存放待插入的结点。但是在移动之前要判断一下当前顺序表是否已满、给定的插入位置是否合法。因此，教材上的 insert_sqlist 算法由 5 个基本操作步骤组成：

1）判断 L 是否已满？当 L.last=maxsize 时，表示 L 已满。

2）判断 i 值是否合法？当 i<1 或 i>L.last 时，i 值不合法。

3）$a_n \sim a_i$ 依次向后移一个位置。

4）将 x 存放到第 i 个位置处。

5）表长加 1。

（2）删除运算。在顺序表中实现删除运算，其核心步骤是从 $a_{i+1} \sim a_n$ 依次向前移一个位置，以使执行删除操作后的顺序表仍然是连续的。但是此处只需判断删除位置 i 是否合法，而无需判断当前顺序表是否为空。对照教材上的 delete_sqlist 算法，它的 3 个基本操作步骤如下：

1）判断 i 值是否合法？当 i<1 或 i>L.last 时，i 值不合法。

2）$a_{i+1} \sim a_n$ 依次向前移一个位置。

3）表长减 1。

（3）定位操作。顺序表上的定位操作比较简单，只要从顺序表的起始处由前往后依次查找值为 X 的结点即可。查找成功，返回它在顺序表中的序号，查找不成功则返回 0。

5. 线性表的链式存储结构——单链表

单链表不同于顺序表，这时用于存放结点的存储单元可能是连续的，也可能是不连续的，因此链表中结点的逻辑次序和物理次序不一定相同。为了反映结点间的逻辑关系，在存放每个结点的同时，必须存储其直接后继的地址，这个地址被称为指针。这样单链表中每个存储结点就由两部分组成：

- 数据域 data：用于存放线性表的数据元素的值。
- 指针域 next：用于存放本结点的直接后继的存放地址。

（1）单链表的形象化表示。下面通过一个例子来说明单链表的实际存储形式是如何被形象化表示的。

设某线性表为（a，b，c，d），4 个元素的实际存储地址依次为 114、136、80、50，假设其数据域和指针域各占 1 个字节，则它们在存储器中的实际存储形式如图 2-1 所示。

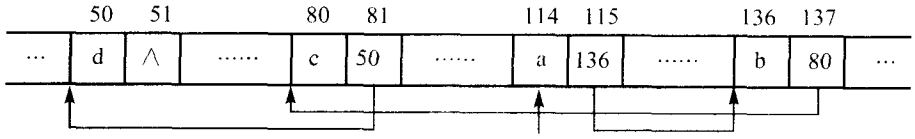


图 2-1　链表的实际存储形式

由图 2-1，在存储一个数据元素的同时，也存储了其直接后继的地址。从图形上看，好像是由该指针域引出了一个指针，指向它的直接后继（如图中箭头所示），为了分析问题的方便，将上述实际存储方式形象化表示为图 2-2 所示的形式。

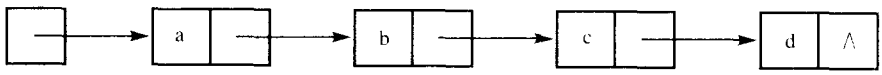


图 2-2　单链表的形象化描述形式

由图 2-2 可知：

1）开始结点无直接前趋，所以应设一个头指针 head 指向开始结点。Head 惟一标识一个链表。

2）终端结点无直接后继，所以其指针域为空（NULL），用∧表示。

3）后一结点的地址存放在前一结点的 next 域中，在表述上等价于前一结点的指针指向后一结点。

（2）区别几个与指针有关的表示形式。设 P、Q 是 pointer 类型的变量（详见教材），请区别下列几种表示及相关的概念。

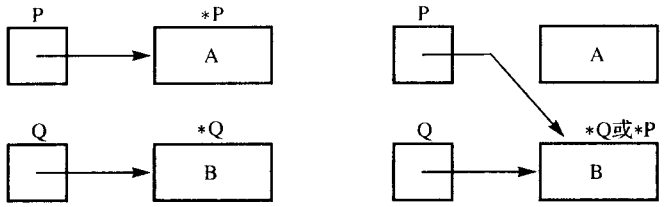
1）pointer：被定义为一指针类型，它所指向的数据类型是 node。在此，node 被定义为由 data 和 next 两个域组成的记录。

2）P、Q：被说明为指针型变量，是静态变量。当其值非空时，它的值是类型为 node 的某个结点的地址（指针）。

3）* P、* Q：表示 P、Q 指针所指向的结点变量，是一动态变量（即通过 malloc(size) 函数得到的存储空间，其名字可以认为就是*P、*Q，能通过 free()函数将它们占用的空间

释放)。

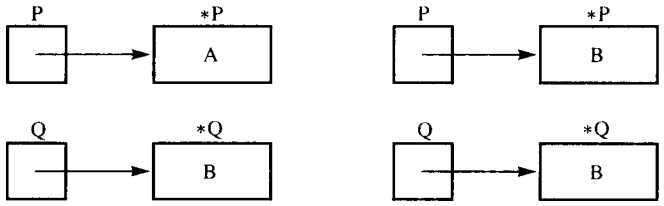
4）P=Q：表示让 P 指向 Q 所指向的结点，如图 2-3 所示。



（a）执行前　　（b）执行后

图 2-3　执行 P=Q 前后的示意图

5）*P=*Q：表示将 Q 所指变量（结点）的值复制到 P 所指的变量中，如图 2-4 所示。



（a）执行前　　（b）执行后

图 2-4　执行*P=*Q 前后的示意图

6. 插入、删除和定位运算在单链表上的实现

（1）插入运算。按照教材上的插入算法 insert-lklist，并配合图 2-5 对单链表上的插入算法进行分析。

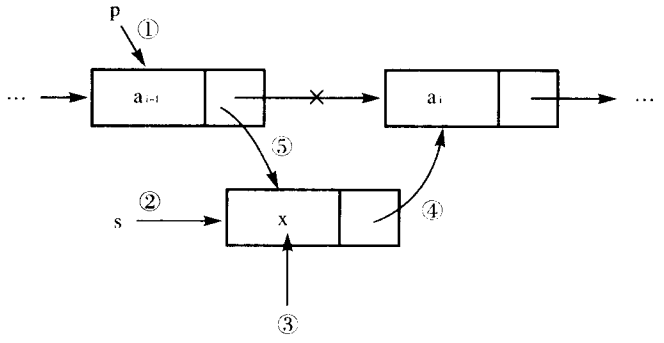


图 2-5　单链表上插入操作基本步骤的示意图

从图中可以看出，整个操作由 5 个基本步骤组成：

1）p=find_lklist(head,i-1)：调用按序查找函数 find_lklist 查找插入位置，使指针 p 指向第 i 个结点的直接前趋。待插入的结点将被插入到 p 所指结点之后。如果 P<>NULL，接着往下执行。

2）s=malloc(size)：调用库函数 malloc，分配一个大小为 size 的连续空间，并将其起始地址存入指针变量 s 中。

3）s->data=x：将待插入结点的值存入 s 所指单元的数据域中。

4）s->next=p->next：p 所指结点的直接后继 a_i 将成为新结点的直接后继。因此，将 a_i 的地址存入 s->next 中。

5）p->next=s：插入后新结点将成为 p 所指结点的直接后继，因此将 s 存入 p->next 中。

由上述分析可知，单链表上的插入操作无需大量移动结点，仅涉及指针的修改。因此，它的时间性能很好。

（2）删除操作。按照教材上的删除算法 delete_lklist，并配合图 2-6 来分析单链表上的删除算法。

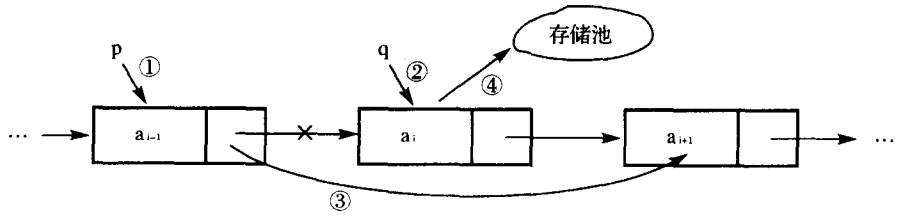


图 2-6　单链表上删除操作的示意图

按图示，整个删除操作由 4 个基本步骤组成：

1）p=find_lklist(hrad,i-1)：调用按序号查找函数 find_lklist 查找删除位置，使指针 p 指向第 i 个结点的直接前趋。

2）q=p->next：保留待删除结点的地址，因为结点删除后，其占用的存储空间要及时释放。

3）p->next=p->next->next 或 p->next=q->next：删除结点后，原来 q 所指结点的直接后继 a_{i+1} 成为 p 所指结点的直接后继，因此将 a_{i+1} 的地址存入 p->next 中。

4）free(q)：调用库函数 free，释放 p 所指的结点变量空间。

由上述分析可知，单链表上的删除操作也无需移动结点，仅涉及指针的修改。

（3）定位运算。单链表上的定位操作比较简单，只要从头部开始顺序查找值等于给定值的结点即可，此处不再赘述。

7. 循环链表和双链表

（1）循环链表。循环链表是首尾相接的链表，其优点是从任一结点出发均可查找到其他结点。实际中，因为许多操作都是在链表尾部进行的，所以用尾指针 rear 取代头指针 head。另外，设置尾指针 rear 也很方便对链表头部的操作。带有尾指针的非空循环链表见教材中的图 2-13（c）所示，注意下列几种表示所代表的含义：

1）*rear：表示尾结点 a_n。

2）*(rear->next)：表示头结点。

3）*(rear->next->next)：表示开始结点 a_1。

（2）双链表。为了方便涉及前趋的操作，在单链表的基础上再添加一个前趋指针 prior，这样就构成了双链表。因为在这种结构中存在两个指针，所以其结构是对称的。设 p 指向某结点，则有下列对称式成立：

p->prior->next=p=p->next->prior

也就是说，结点*p 的地址既存放在其直接前趋的后继指针域中，又存放在其直接后继的前趋指针域中。所以在双链表特别是双循环链表上进行插入、删除操作都很方便。

8. 顺序表与链表的比较

（1）顺序表的优缺点。

优点：

1）空间利用率高。

2）可以实现随机存储。

缺点：

1）插入、删除操作中要大量移动结点，时间性能差。

2）需要事先确定顺序表的大小，估计小了会发生溢出现象，估计大了又将造成空间的浪费。

（2）链表的优缺点。

优点：

1）插入、删除操作中无需移动结点，时间性能好。

2）因为是动态分配存储空间，所以只要有空闲空间就不会发生溢出现象。

缺点：因为每个结点都要额外设置指针域，因此空间利用率低。

通过上述比较，可以看出算法的时间性能与空间性能往往是一对矛盾，时间性能的改善要以牺牲空间性能为代价，反之亦然。因此在实际中，要根据具体情况来确定采用哪种存储结构。

9. 串的基本概念

串是一种特殊的线性表，它的每个结点仅含一个字符，教材上给出了主串、子串、空串、串长等概念的定义。

1）空串是指长度为零的串，其中不含任何字符。而空格串则是指含有空格（空格也是字符）的串，因此它的长度不为零。

2）空串是任意串的子串，任意串是其自身的子串。

串的基本运算有以下几种

1）ASSIGN(S,T)：将串 T 的值赋给串 S。

2）EQUAL(S,T)：判断两串是否相等，当 S=T 时返回值为 1，当 S≠T 时返回值为 0。

3）LENGTH(S)：求串 S 的长度。

4）CONCAT(S,T)：将串 S 和串 T 连接在一起构成一个新串。

5）SUBSTR(S,i,j)：从串 S 的第 i 个字符起连续取出 j 个字符构成一个子串。

6）INSERT（S1,i,S2)：将串 S2 插入到 S1 的第 i 个字符之后。

7）DELETE(S,i,j)：从串 S 的第 i 个字符起连续删除 j 个字符。

8）INDEX(S,T)：当串 S 中存在与串 T 相等的子串时，则返回第一个子串 T 在串 S 中的第一个字符的序号，否则返回 0。

9）REPLACE(S,T,R)：用子串 R 替换串 S 中所有子串 T。

串的存储方式最常见的是顺序存储方式和链式存储方式，分别称其为顺序串和链串。对于按字编址的计算机来讲，顺序串又可以分为紧缩格式和非紧缩格式两种。前者每个存储单元（一个字）里存放多个字符，而后者每个存储单元（一个字）只存放一个字符。

2.2　配套教材习题分析与解答

1．叙述以下概念的区别：头指针变量、头指针、头结点、首结点，并说明头指针变量和头结点的作用。

【解答】头指针变量：该变量的值是指向单链表的第一个结点的指针。因此，它是用于存放头指针的变量。

头指针：是指向单链表的第一个结点的指针。

首结点：是指用于存储线性表中第一个数据元素的结点。

头结点：在链表的首结点之前附设的一个结点，称为头结点。

头指针变量的作用：对单链表中任一结点的访问必须首先根据头指针变量中存放的头指针找到第一个结点，再依次按各结点链域存放的指针顺序往下找，直到找到或找不到。头指针变量具有标识单链表的作用，所以用头指针变量来命名单链表。

头结点的作用：该结点的数据域中不存储数据元素，其作用是为了对链表进行操作时，将对第一个结点的处理和对其他结点的处理统一起来。

2．有哪些链表可仅由一个尾指针来惟一确定，即从尾指针出发能访问到链表上任何一个结点。

【解答】循环单链表和循环双链表。

3．设 A=（$a_1,a_2,...,a_n$）和 B=（$b_1,b_2,...,b_m$）是两个线性表（假定所含数据元素均为整数）。若 n=m 且 $a_i=b_i$（i=1,...,n），则称 A=B；若 $a_i=b_i$（i=1,...,j）且 $a_{j+1}<b_{j+1}$（$j<n\leqslant m$），则称 A<B；在其他情况下均称 A>B。试编写一个比较 A 和 B 的算法，当 A<B、A=B 或 A>B 时分别输出-1、0 或 1。

【分析】根据题目要求，从两个表中第一个数开始比较，若两数相等，继续比较下面两数，直到两表同时查找完毕，或者遇到两表中有一个表先查找完毕，或者两数不相等。由不同情况，根据题目要求，返回不同值。

【解答】

```
intcomp (sqlist a,b)                    /*a 和 b 均为单链表*/
{ i=1;
```